International Journal of Computer Science and Business Informatics

IJCSBI.ORG

Efficient Numerical Integration and Table Lookup Techniques for Real Time Flight Simulation

P Lathasree
CSIR-National Aerospace Laboratories
Old Airport Road, PB No 1779, Bangalore-560017

Abhay A Pashilkar
CSIR-National Aerospace Laboratories
Old Airport Road, PB No 1779, Bangalore-560017

ABSTRACT
A typical flight simulator consists of models of various elements such as the flight dynamic model, filters and actuators, which have fast and slow eigen values in the overall system. This results into an electromechanical control system of stiff ordinary differential equations. Stability, accuracy and speed of computation are the parameters of interest while selecting numerical integration schemes for use in flight simulators. Similarly, accessing huge aerodynamic and engine database in table look-up format at high speed is an essential requirement for high fidelity real time flight simulation. A study was carried out by implementing well known numerical integration and table lookup techniques in a real time flight simulator facility designed and developed in house. Table lookup techniques such as linear search and index computation methodology using novel Virtual Equi-Spacing concept were also studied. It is seen that the multi-rate integration technique and the table look up using Virtual Equi-Spacing concept have the best performance amongst the techniques studied.

Keywords
Real-Time Flight Simulation, Aerodynamic and Engine database, Virtual Equi-Spacing concept, table look up and interpolation, Runge-Kutta integration, multi-rate integration.

1. INTRODUCTION
Flight simulation has a vital role in the design of aircraft and can benefit all phases of the aircraft development program: the early conceptual and design phase, systems design and testing, and flight test support and envelope expansion [1]. Simulation helps in predicting the flight behavior prior to flight tests. It helps in certification of the aircraft under demanding scenarios. Flight Simulation is widely used for training purposes in both fighter and transport aircraft programs [2]. Therefore, Modeling & Simulation is one of the enabling technologies for aircraft design.

The fidelity of the simulation largely depends on the accuracy of the simulation models used and on the quality of the data that goes into the model. A faithful simulation requires an adequate model in the form of

mathematical equations, a means of solving these equations in real-time and finally a means of presenting the output of this solution to the pilot by means visual motion, tactile and aural cues [3].

The Real-Time Flight Simulator implies the existence of a Man-In-the-Loop operating the cockpit controls [4]. Because of the presence of the pilot-in-the-loop, the digital computer executing the flight model in the simulator must solve the aircraft equations of motion in 'real-time' [5]. Real-Time implies the occurrence of events at the same time in the simulation as seen in the physical system. All the associated computations should be completed within the cycle update time [6].

The basis of a flight simulator is the mathematical model, including the database package, describing the characteristic features of the aircraft to be simulated. The block schematic of flight simulator is shown in Figure 1 with constituent modules such as aerodynamic, engine, atmosphere (static and dynamic), actuator etc. The simulation model for atmosphere includes the static and dynamic atmosphere components. Dynamic atmosphere model caters for turbulence, wind shear and cross wind. Dryden and Von Karman models are generally used for the simulation of atmospheric turbulence [7].

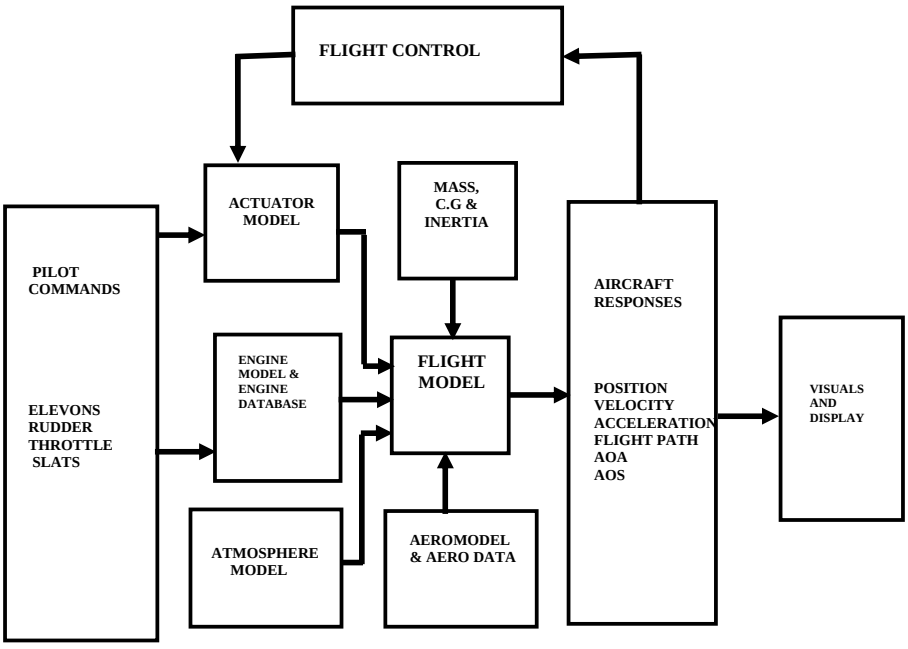

Figure 1 Block Schematic of Flight Simulation

Mathematical models, used to simulate modern aircraft, consist of a set of non-linear differential equations with large amounts of aerodynamic function data (tables), sometimes depending on 4 to 5 independent variables. These aerodynamic data tables result in force and moment coefficients which contribute to the total forces and moments. The equations of motion are dependent on these forces and moments. They are solved by the digital computer using a suitable numerical integration algorithm. This allows the designer to create the complete range of static and dynamic aircraft operating conditions, including landing and takeoff [6].

The type of method used for the integration of ordinary differential equations is critical for real time simulation. The choice of an integrating algorithm is a trade-off between simplicity, which affects calculation speed, and accuracy. Also, real simulation needs high speed data access. The aerodynamic and engine database used for real-time simulation are huge and complex. Hence, the types of table look-up methods used for access of data from aerodynamic and engine database also become critical.

This paper discusses the efficient table look up and interpolation schemes and numerical integration techniques which can be used for ensuring accurate real-time computations in a flight simulator.

2. REVIEW OF EXISTING TECHNIQUES

The existing numerical integration techniques and table lookup and interpolation methods for real time implementation are discussed in this section.

2.1 Numerical Integration

Many linear numerical integration techniques with single and multi step are available which can also be classified into implicit and explicit numerical integration techniques [8]. With respect to the stability and accuracy, each of these numerical integration techniques has advantages and disadvantages [8]. Depending on the performance, these methods can be suitably used for stiff and non-stiff systems. Methods not designed for stiff problems must use time steps small enough to resolve the fastest possible changes, which makes them rather ineffective on intervals where the solution changes slowly. The most popular numerical integration methods are listed below.

- Taylor Series Methods
- Runge-Kutta Methods
- Linear Multi Step Methods
- Extrapolation methods

The linear multistep methods (LMMs) require past values of the state. They are therefore not self-starting and do not directly solve the initial-value problem [9]. The simplest Runge-Kutta (RK) method is Euler integration,

which merely truncates the Taylor series after the first derivative and is very accurate [9]. An RK method (e.g., Euler) could be used to generate the starting values for LMMs.

Higher order RK algorithms are an extension Taylor series expansion to higher orders. An important feature of the RK methods is that the only value of the state vector that is needed is the value at the beginning of the time step; this makes them well suited to the Ordinary Differential Equations initial value problem [1].

2.1.1 Stability, Accuracy and Speed of Computation

While choosing the numerical integration technique, one frequently has to strike a compromise between three aspects [10-11].

- Speed of the method
- Accuracy of the method
- Stability of the method

Speed of the method becomes an essential feature especially for real time simulation.

Accuracy of the method is also an important aspect and needs to be considered when choosing a method to integrate the equations of motion [12]. Accuracy of the numerical integration technique can be determined from step size, number of steps to be executed and truncation error terms [10-11]. Generally, two types of errors will be introduced by the numerical integration methods viz. round-off errors and discretisation errors. Round-off errors are a property of the computer and the program that is used and occur due to the finite number of digits used in the calculations [13-14]. Discretisation/ truncation errors are property of the numerical integration method.

Stability can be defined as the property of an integration method that keeps the integration errors bounded at subsequent time steps [12]. An unstable numerical integration method will make the integration errors grow exponentially resulting in possible arithmetic overflow just after a few time steps.

Stability of numerical integration technique generally depends on the system dynamics, step size and order of the chosen technique and is harder to assess [10-11]. Impact of numerical integration method in terms of stability can be assessed by applying it to a well-conditioned differential equation and then investigating the limits of the onset of instability [10-11]. In the context of stability of numerical integration, it is understood that a stable continuous system results in a stable discrete-time system. Numerical stability is important for fixed-step Runge-Kutta integrators because of the limitations imposed on the integration step size. Generally, selection of the integration

step size will be carried out based on analysis on the stability of the numerical integration technique. [15]. Numerical stability will be an issue when the chosen integration step size produces z-plane poles close to the Unit Circle.

If the poles are located inside the Unit circle, then the system will be stable. Increasing T (step size) eventually causes one of the z-plane poles to be on the Unit Circle where the system becomes marginally stable. Depending on the location of λT (product of characteristic root and step size) on the stability boundary of respective integrator, it is possible to estimate the maximum allowable integration step size (Tmax) for the system solution to be at least marginally stable. Beyond Tmax, the system solution will become unstable. Hence, it is very essential to consider stability boundaries for different numerical integrators while selecting the integration step size. Figure 2 shows the stability boundaries for Runge-Kutta methods [15].

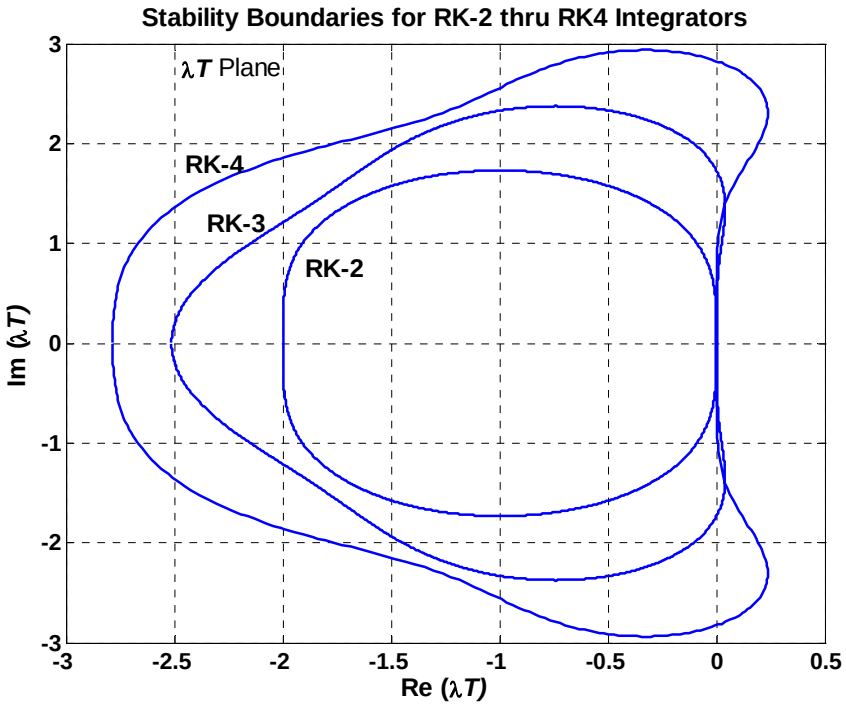

Figure 2 Stability Boundaries for Runge-Kutta methods

2.1.2 Numerical Integration techniques for Stiff Systems

'Stiffness' of the differential equations may be defined as the existence of one or more fast decay processes in time, with a time constant that is small compared to the time-span of interest [13].

One has to consider the following two points while choosing the numerical integration technique [10-11]:

- The integration technique should be chosen such that any error it introduces is small in comparison to the errors associated with the main terms of the model equations;
- The numerical integration techniques should be able to solve the system of differential equations within the real-time frame rate.

Many integration techniques, for non-real time simulation applications, are available that work well with the stiff systems [16-17]. Two approaches that can be used for simulating stiff systems with respect to real time and non-real time simulation will be discussed here. The first approach considers selection of numerical integration technique that works well in the presence of stiffness.

The second approach involves the use of multi-rate integration to simulate stiff systems. In multi-rate simulations, the simulation is split into multiple tasks that are executed with different integration step times. The inverse of the integration step time is termed as frame rate and expressed in frames per second. This multi-rate integration technique is useful for real-time applications as well as non real-time applications.

Of the two approaches discussed for the simulation of stiff systems, only the multi-rate integration technique is applicable for real time applications.

The control systems with electrical and mechanical components, referred as electromechanical control systems, are composed of fast and slow subsystems. Generally, the mechanical systems being controlled are much slower when compared to the components in electronic controllers and sensors. This results in an electromechanical control system with fast and slow dynamics. The aircraft pitch control system is an example of system of stiff ordinary differential equations comprising of aircraft dynamics and actuators [15].

Kunovsky et al have established the need of multi-rate integration for real time flight simulation [18] with an example of aircraft pitch control system comprising of slow aircraft dynamics and fast actuator dynamics using Runge-Kutta and Adams-Bashforth numerical integration techniques. The airframe module of aircraft pitch control system is modeled as a linear second-order system to account for the short-period longitudinal dynamics. Generally, selection step size for numerical integration will be carried out based on the analysis of stability and dynamic accuracy. Ts and Tf are the integration step sizes of slow and fast systems respectively.

The numerical integrator used to update slow system is termed as ‘‘master’’ routine, and the integration method used to update the fast system is called

as ‘‘slave’’ routine. It is common to use conventional numerical integration schemes such as Runge-Kutta methods for both ‘master’ and ‘slave’ systems. For the example studied here, the multi-rate integration scheme with RK-4 is chosen for master and slave routines. The implementation is carried out in the Matlab environment. For a pitch command of 2deg, simulation is carried for the state space based simulink model. This result is compared with the analytical solution and the response obtained using a multi-rate integration scheme. The comparison of theta and elevator responses for three methods is shown in Figure 3 and Figure 4 respectively.

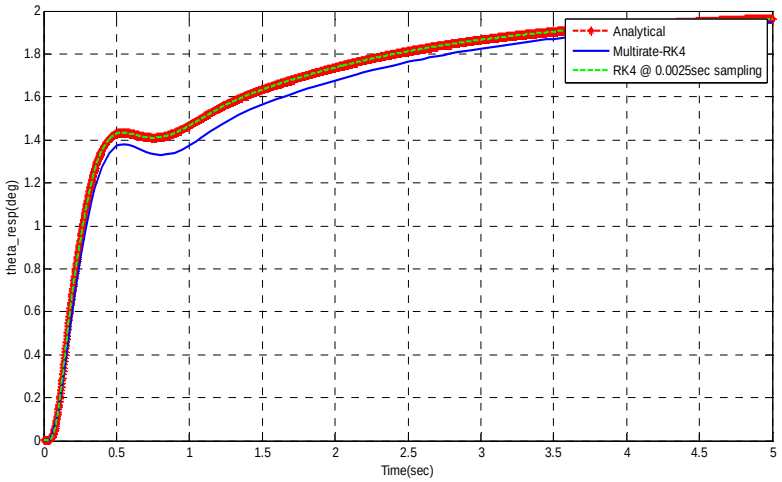

Figure 3 Comparison of Theta response

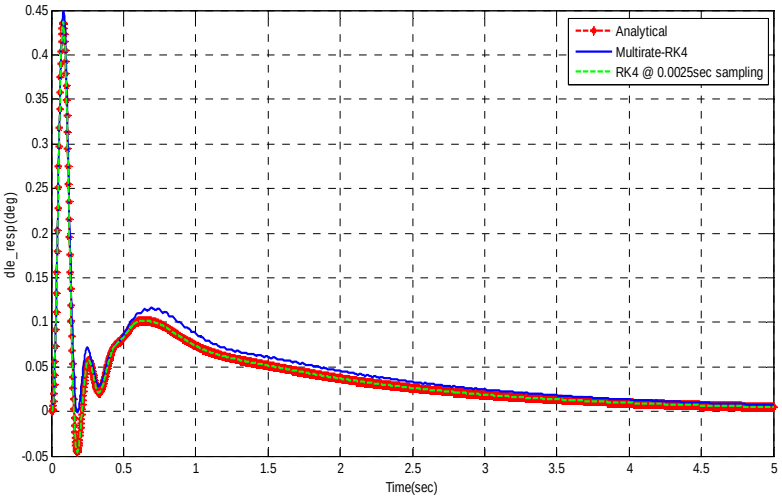

Figure 4 Comparison of elevator response

The responses obtained from analytical solution are taken as the reference. From the figures, it can be seen that the response obtained using simulink model at step size 0.0025 is matching well with reference whereas the response obtained using the multi-rate integration exhibits loss of accuracy. The multi-rate integration scheme would be recommended for real time simulation even though there is some loss of accuracy, since the smaller step size may deteriorate the performance.

2.2 Table look-up and Interpolation

Generally, an index search or look-up process will be performed first to locate the data and this is followed by linear interpolation. Following steps need to be performed for table look-up process [3]:

1. First we should decide between which pair of values in the table the current input value of independent variable (X) lies
2. Next, calculate the local slope
3. Finally, apply the linear interpolation formula

For real-time simulation, it is always important to save the processing time. One of the techniques to save the processing time is to remember the index of the lower pair the interpolation range used in the previous iteration. The value of the independent variable (X) is unlikely to have changed substantially from one time step to the next, and hence it is a good first try to use the same interval as before and thus save time in searching from one end of the table each time.

Huge and complex aerodynamic and engine database has to be handled in such a way that it can be easily read and interpolated for a given set of input conditions. One way of ensuring the speed required for real-time simulation, is to have uniformly spaced database. For this, the normal practice is to convert the supplied database with non-uniform break points for independent variables to equi-spaced format. It is necessary to choose an appropriate step size for independent variables such as Angle of Attack, Mach number, Elevator, Angle of Sideslip, Power Lever Angle (PLA) etc to convert this non-uniform database to equi-spaced format. This is normally termed as conventional equi-spacing concept. We propose a new concept called Virtual Equi-Spacing where the original database with non-uniform break points is retained. With the assumption of virtual equi-spacing, the search process can be eliminated [19] as the index is directly computed. The computation of index in Virtual Equi-Spacing concept is explained in the following section.

2.2.1 Virtual Equi-Spacing Concept

A novel method is proposed which would retain original data with unevenly spaced break points and satisfies real-time constraint without loss of accuracy. In this method, an evenly spaced breakpoint array that is a

superset of the unevenly spaced break points will be created for the independent variables and shall be referred as 'Address Map'. The index into this evenly spaced array can be directly computed (Refer Figure 5). This index is then used in an equivalent breakpoint index array that provides pointers to the appropriate interpolation equation.

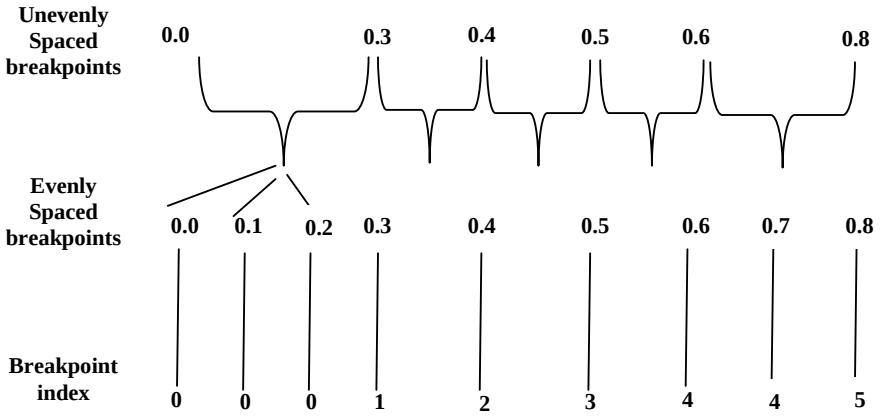

Figure 5 Indexing scheme in Virtual Equi-Spacing Concept

The Virtual Equi-Spacing concept satisfies real-time speed constraint without loss of accuracy for the real time flight simulators. This technique eliminates search process and directly computes the index of data tables. The Virtual Equi-Spacing concept works as follows. A division of the desired input value by the step size chosen for the creation of Address map table gives the location 'K'. The value of address map [K], say 'i' is used as a pointer in data table to get the final data component value i.e. Table[i] for the desired input. This is now demonstrated with a typical example.

Aircraft engine database is a three dimensional dataset where thrust is a function of three independent variables viz. Mach number, PLA and altitude. The technique of computing index values in address maps and the index values in data arrays is explained with PLA dimension.

Let pla_val = 54.0 deg for the Mach number 0.4 and Altitude 4500.0m. The PLA and Thrust relationship at these conditions is given in Table 1.

The computation of index values and thereby data values is presented in Appendix along with the pseudo code. The engine database of a high performance fighter aircraft is used to demonstrate the table look-up and interpolation schemes. This database consists of engine parameters such as thrust, specific fuel consumption, N1 rpm, N2 rpm etc. supplied as function

of Mach number, PLA and altitude. This index computation methodology using Virtual Equi-Spacing concept is extended to multi-dimension tables of wind tunnel database.

Table 1 PLA and Thrust relationship

PLA(deg)	Thrust(kN)
28.	-0.63
42.	3.21
54.	8.7
66.	13.81
78.	20.24
90.	26.32
104.	28.09
107.	30.26
130.	44.84

The next section presents a study on efficient table look up algorithms and numerical integration algorithms suitable for real time implementation in flight simulators.

3. RESULTS

From the survey of existing techniques for numerical integration and table look-up, concept of multi-rate integration and Virtual Equi-Spacing concept are implemented for real-time flight simulation and studied. This implementation is carried out in the real-time flight simulation facility designed and developed at CSIR-NAL.

Figure 6 shows the conceptual flowchart of real time flight simulation. The simulation is typically started from an equilibrium / trim condition. For the given set of pilot inputs, flight dynamic module solves the equations of motion using the chosen numerical integration method. It is necessary that all the associated computations should be completed within the cycle update time for real-time simulation. These computations are completed ahead of cycle update time and the beginning of the next cycle is delayed till the internal clock signals the next cycle update as shown in Figure 6.

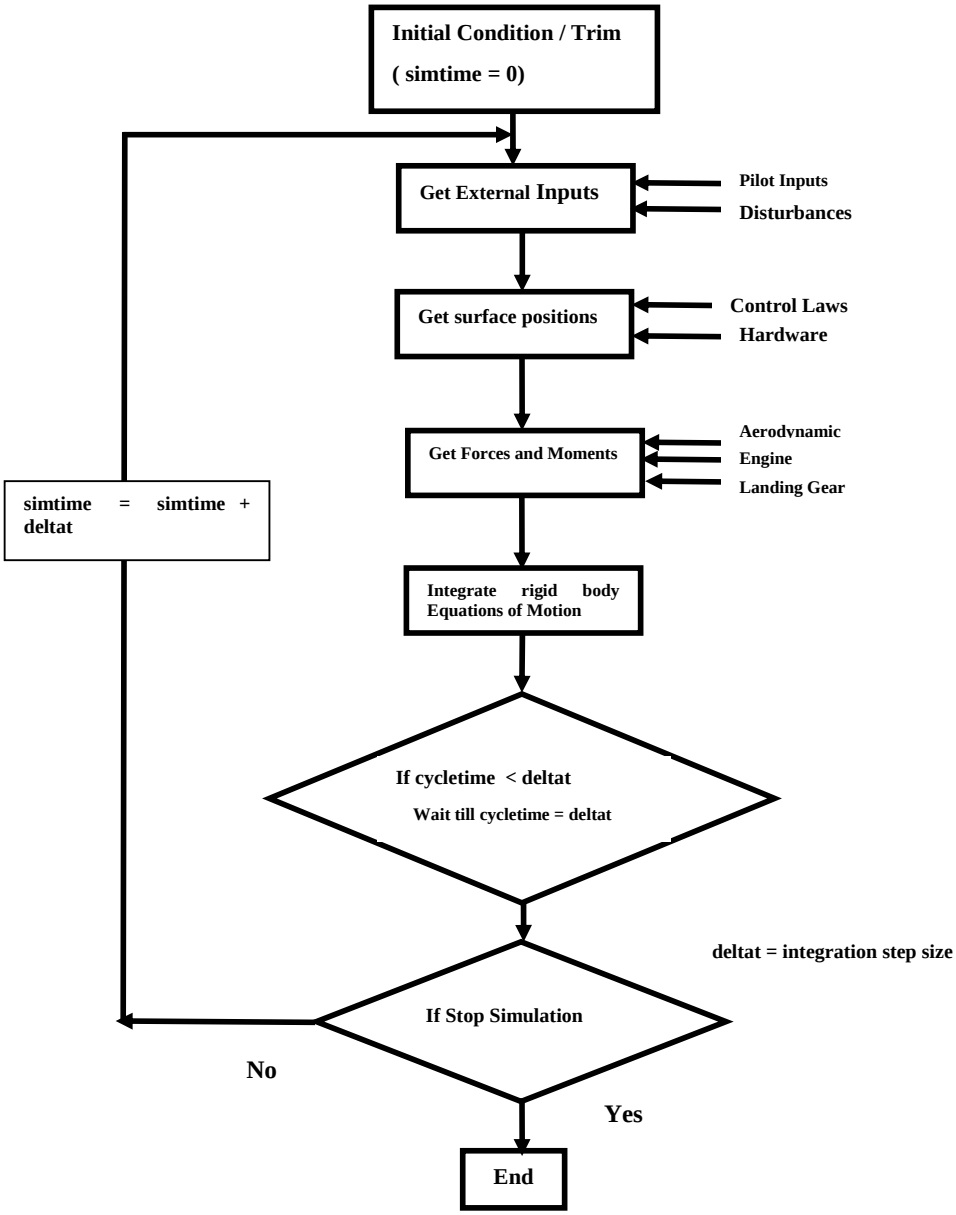

Figure 6 **Conceptual flowchart of real-time flight simulation**

3.1 Timing Analysis

The timing analysis is carried out for the numerical integration and table loop up techniques and the results are presented.

3.1.1 Numerical Integration

The concept of multi-rate integration is adopted for the real-time flight simulation facility designed and developed at CSIR-NAL. The full nonlinear model of the aircraft dynamics along with the actuator dynamics for a light transport aircraft is considered for this real-time flight simulation environment. The aircraft dynamics of light transport aircraft constitute the slow dynamics and fast dynamics is composed of actuator dynamics. The nominal integration step size of 0.025sec is chosen for the airframe simulation purpose. Similarly, for the actuator dynamics 0.0025sec is chosen as integration step size based on the analysis of stability and dynamic accuracy. It can be seen that the ratio of step sizes of slow system to fast system (frame ratio) is 10 indicating a stiff system. The multi-rate integration scheme with frame ratio 10 and simulation cycle update time 0.025sec ensures the handling of slow and fast subsystems. The Runge-Kutta pair of Bogacki and Shampine [20] is currently being used for the numerical integration of slow and fast dynamics. Table 2 presents the timing analysis for the simulation (off-line) carried out using the windows based timer function with the resolution in micro seconds.

Table 2 Timing analysis for multi-rate and mono-rate integration techniques

Duration of Simulation	Description	Time (sec)
35sec	Multi-rate integration with ts = 0.025 & tf = 0.0025	0.4271
	Mono-rate integration with deltat 0.0025sec	1.8518
50sec	Multi-rate integration with ts = 0.025 & tf = 0.0025	1.1058
	Mono-rate integration with deltat 0.0025sec	2.7181
100sec	Multi-rate integration with ts = 0.025 & tf = 0.0025	1.1077
	Mono-rate integration with deltat 0.0025sec	4.9378

Figure 7 shows the plots of aircraft responses obtained with pitch stick doublet for mono-rate integration with 0.0025sec sampling time and multi-rate integration with 0.025 / 0.0025sec sampling times. From the plots, it can be seen that the mismatch between the multi-rate integration scheme and the mono-rate solution is negligible.

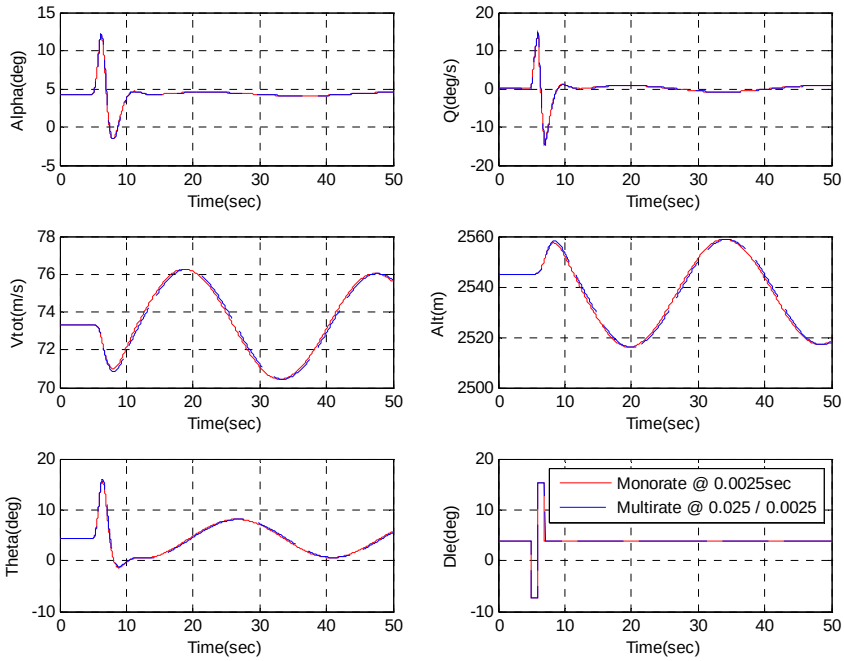

Figure 7 **Comparison plots for aircraft response variables with mono-rate and multi-irate integration schemes**

For real time applications, accuracy is the feature that must be sacrificed in conflicts with other properties. It is better to obtain a solution with some small error than not be able to obtain it at all in the allowed time. Moreover, many real time applications incorporate a feedback control. Feedback control helps to compensate errors and disturbances, including integration errors. For real time flight simulation, the multi-rate integration scheme may be adopted for better computational time.

3.1.2 Table Look-Up and Interpolation

This flight simulator facility is using the aerodynamic and engine database with unevenly spaced break points. It is proposed to use the original dataset with unevenly spaced breakpoints and facilitate a faster table look up and interpolation.

As already discussed, a technique to save time is to remember the index of the lower pair of the interpolation range used last time. From one time step to the next, the value of the independent variable X is unlikely to have changed substantially and so it would be a good first try to use the same interval as before and thus avoid waste of time in searching from one end of the table each time. Hence, linear search with option of remembering previous used index is used for the timing analysis.

Timing analysis is carried out for linear search with option of remembering previously used index and the novel Virtual Equi-Spacing concept proposed in the previous section. A windows based timer function with the resolution in micro seconds is used to obtain the time taken for the table look up and interpolation. Generally, this process includes, computing the location of data component value in the corresponding data table and interpolation.

Table 3 Timing studies for different search and interpolation techniques

Mach number 0.4 Altitude 4500m	**Time in Micro seconds**	
	Linear Search (with option of remembering previous used index)	**Virtual Equi-Spacing concept**
PLA / 50	18.08	12.65
PLA / 90	18	12.5
PLA / 107	17.2	12.75
PLA / 110	19.1	12.44

The recommended Virtual Equi-Spacing technique has been used for the table lookup and interpolation of the aerodynamic and engine database consisting of around two lakh data points (representing a high performance fighter aircraft). The engine data base of size 20000 data points is taken as an example to carry out the study. Table 3 gives the timing of two different techniques studies at different PLA conditions while Mach number and altitude are maintained same. From the table, it is found that Virtual Equi-Spacing technique takes lesser time. The accuracy is maintained as the actual data tables are not affected.

4. CONCLUSIONS

A study was carried out to recommend efficient numerical integration and table look up techniques suitable for real time flight simulation comprising of system of stiff ordinary differential equations. Numerical integration and table lookup techniques available in literature were implemented in a real time flight simulator facility designed and developed in house. Aircraft pitch control system representing the slow and fast subsystems was considered for the study on numerical integration techniques. Table lookup techniques such as linear search and index computation methodology using Virtual Equi-Spacing concept have been studied for an example of the engine database of a high performance fighter aircraft. The Virtual Equi-Spacing is a new

concept developed for interpolation of large multi-dimensional tables frequently used in flight simulation.

With excessively small step size, it is possible to solve the stiff differential equations, but this results in performance penalty, an important aspect of real time simulation. Hence, it is recommended to opt for multi-rate simulation, where it is necessary to use a step size for the actuator simulation that is sufficiently small to ensure an accurate and stable actuator solution and a larger step size for simulating the slower dynamics of the airframe.

The Virtual Equi-Spacing concept for table lookup and interpolation leads to faster and accurate data access, an essential feature of real-time simulation while handling larger databases.

From the results, it is found that the recommended multi-rate integration technique and the table look up using Virtual Equi-Spacing concept perform better.

5. ACKNOWLEDGMENTS

The authors would like to thank Mr Shyam Chetty, Director, CSIR-NAL and Dr (Mrs) Girija Gopalratnam, Head, Flight Mechanics and Control Division, CSIR-NAL for their guidance and support.

REFERENCES

[1] Ken A Norlin, Flight Simulation Software at NASA Dryden Flight Research Center, NASA TM 104315, October 1995

[2] David Allerton, Flight Simulation- past, present and future, The Aeronautical Journal, Vol 104, Issue No. 1042, pp 651-663, December 2000

[3] J M Rolfe and K J Staples, Flight Simulation, Cambridge University Press, Year of publication 1991

[4] Flight Mechanics & Control Division, CSIR-National Aerospace Laboratories, NAL-ASTE Lecture Series, May 2003

[5] Max Baarspul, A review of Flight Simulation Techniques, Progress in Aerospace Sciences, (An International Review Journal), Vol. 27, Issue No. 1, pp 1-120, March 1990

[6] Joseph S. Rosko, Digital Simulation of Physical systems, Addison-Wesley Publishing Company. Year of publication 1972

[7] Beal, T.R., Digital simulation of atmospheric turbulence for Dryden and Von Karman models, Journal of Guidance Control and Dynamics, Vol 16, Issue No. 1, pp132–138, February 1993.

[8] http://qucs.sourceforge.net/tech/node24.html Accessed on 8.1.2014

[9] Brian L Stevens and Frank L Lewis, Aircraft and Control and Simulation, John Wiley & Sons Inc. Year of Publication 1992

[10] David Allerton, Principles of Flight simulation, John Wiley & Sons Ltd. Year of Publication 2009

[11] http://www.scribd.com/doc/121445651/PRINICIPLES-OF-FLIGHT-SIMULATION Accessed on 8.1.2014

[12] http://mat21.etsii.upm.es/mbs/bookPDFs/Chapter07.pdf, Numerical Integration of Equations of Motion Accessed on 26.6.2012

[13] Marc Rauw, FDC 1.4 – A SIMULINK Toolbox for Flight Dynamics and Control Analysis, Draft Version 7, May 25, 2005

[14] John W Wilson and George Steinmetz, Analysis of numerical integration techniques for real-time digital flight simulation, NASA-TN-D-4900 dated November 1968, Langley Research Center, Langley Station, NASA, Hampton, VA

[15] Harold Klee and Randel Allen, Simulation of Dynamic Systems with Matlab and Simulink, Second Edition, CRC Press, Taylor and Francis Group, 2011

[16] Jim Ledin, Simulation Engineering:Build better embedded systems faster, CMP books, Publication Year 2001

[17] http://www.embedded.com/design/real-world-applications/4023325/Dynamic-System-Simulation

[18] Ji˘r´ı Kunovsk´y et al, Multi-rate integration and Modern Taylor Series Method, Tenth International conference on Computer Modeling and simulation, 2008, IEEE Computer Society.

[19] Donald E. Knuth, The art of computer programming – Volume 3 / Sorting and Searching, Addison-Wesley Publishing Company. Year of publication 1973

[20] http://en.wikipedia.org/wiki/Bogacki%E2%80%93Shampine_method Accessed on 12/10/2008

Appendix

Computing index values and data values:

```
data pladata /
    *  28.0,42.0,54.0,66.0,78.0,90.0,104.0,107.0, 130/
```

The Address Map assumes the virtual equi-spaced data with 1.0deg step. For the PLA value 28 to 41, the index number will be 1. For the PLA value 42.0 to 53.0, the index number will be 2. Similarly, for the PLA value 54.0 to 65.0, the index number will be 3 and so on.

```
 data (plamap(it), it=1,103) /
    *  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    *  2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
    *  3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
    *  4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
    *  5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
    *  6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
    *  7, 7, 7,
    *  8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
    *  8, 8, 8, 8, 8, 8, 8, 8, 8,
    *  9 /
```

The index into these address maps can directly computed based on the step size.

```
iplav = int((pla_val-28.0)/1.0) + 1  = 27
iplax = plamap(iplav)      =  3
```

Based on this index number corresponding to the independent variable PLA, it is possible to obtain thrust value in the table.

```
thrust_val11 =  thrust_tab(iplax) =  8.7
thrust_val12 =  thrust_tab(iplax+1) = 13.81
thrust_val  =    thrust_val11  +  ((thrustval12-thrustval11)/(pladata(iplax+1)-
pladata(iplax))*pla_val-pla_data(iplax)  = 8.7
```

If PLA value is lying between two break points e.g. pla_val = 70.5

```
iplav = int((60.5-28)/1) + 1 = 43
iplax = plamap(iplav) = 4
thrust_val11 =  thrust_tab(iplax) =  13.81
thrust_val12 =  thrust_tab(iplax+1) = 20.24
thrust_val  =    thrust_val11  +  ((thrustval12-thrustval11)/(pladata(iplax+1)-
pladata(iplax))*pla_val-pla_data(iplax)  = 16.2213
```